VŠB – Technická univerzita Ostrava
Fakulta elektrotechniky a informatiky
Katedra informatiky

Dashboard pro přehledné zobrazování dat firmy

Dashboard for Well-Arranged Displaying Company Data

2017 **Vojtěch Patschka**

VŠB - Technická univerzita Ostrava
Fakulta elektrotechniky a informatiky
Katedra informatiky

Zadání bakalářské práce

Student: **Vojtěch Patschka**

Studijní program: B2647 Informační a komunikační technologie

Studijní obor: 2612R025 Informatika a výpočetní technika

Téma: Dashboard pro přehledné zobrazování dat firmy
Dashboard for Well-Arranged Displaying Company Data

Jazyk vypracování: čeština

Zásady pro vypracování:

Cílem bakalářské práce je vytvořit klienta informačního systému K2 pro zařízení s operačním systémem Windows 10 (Windows Universal Platform). Aplikace bude dostupná ve Windows Store a dokáže zobrazit a vyčíslit dashboardy definované v informačním systému. Dále pak umožní uživatelům přijímat notifikace ze systému.

1. Studium technologie Windows Universal Platform.
2. Studium API informačního systému K2.
3. Implementace autentizace K2 API.
4. Implementace tenkého rest klienta (zobrazení tabulky dashboardů).
5. Implementace grafické komponenty zobrazující dashboard.
6. Implementace komponent pro filtrování (slicer, combo, MultiSelectCombo, TreeFilter).
7. Interakce mezi reporty.
8. Tile navigace mezi dashboardy.
9. Implementace notifikací.

Seznam doporučené odborné literatury:

Podle pokynů vedoucího bakalářské práce.

Formální náležitosti a rozsah bakalářské práce stanoví pokyny pro vypracování zveřejněné na webových stránkách fakulty.

Vedoucí bakalářské práce: **Ing. Tomáš Szkandera**

Konzultant bakalářské práce: doc. Mgr. Jiří Dvorský, Ph.D.

Datum zadání: 01.09.2016

Datum odevzdání: 28.04.2017

doc. Dr. Ing. Eduard Sojka
vedoucí katedry

prof. RNDr. Václav Snášel, CSc.
děkan fakulty

Prohlašuji, že jsem tuto bakalářskou/diplomovou práci vypracoval samostatně. Uvedl jsem všechny literární prameny a publikace, ze kterých jsem čerpal.

V Ostravě dne 26.4.2017 .....

Souhlasím se zveřejněním této bakalářské/diplomové práce dle požadavků čl. 26, odst. 9 Studijního a zkušebního řádu pro studium v bakalářských/magisterských programech VŠB-TU Ostrava.

V Ostravě dne 26.4.2017 .....

Abstrakt

Cílem této bakalářské práce je vytvoření funkčního klienta informačního systému pro operační systém Windows 10 pomocí technologie Universal Windows Platform. Aplikace má za úkol přehledně zobrazit data pomocí grafů a tabulek, a jejich filtraci pomocí několika druhů filtrů. Data jsou stahována z webových služeb společnosti K2 atmitec. Dále program umožňuje stahování notifikací a příjem push notifikací systému Windows.

Klíčová slova

Dashboard, Aplikace, API, JSON, Graf, Tabulka, Notifikace

Abstract

The purpose of this bachelor thesis is creation of functional client of information system for operation system Windows 10 with Universal Windows Platform technology. The task of application is to show data in well-arranged manner using charts and tables, and filtration of data by several types of filters. Data are downloaded from web services created by company K2 atmitec. Another task of application is to download notification and allow receiving of push notification by Windows system.

Key words

Dashboard, Application, API, JSON, Chart, Table, Notification

Obsah

Seznam použitých zkratek

API – Application Programming Interface

ASP – Active Server Pages

CLR – Common Language Runtime

GUI – Graphical User Interface

GUID – Globally Unique Identifier

HMAC – Keyed-hash Message Authentication Code

JSON – JavaScript Object Notation

MD5 – Message-Digest algorithm

UML – Unified Modeling Language

URI – Uniform Resource Identifier

URL – Uniform Resource Locator

UWP – Universal Windows Platform

XAML – Extensible Application Markup Language

Seznam ilustrací

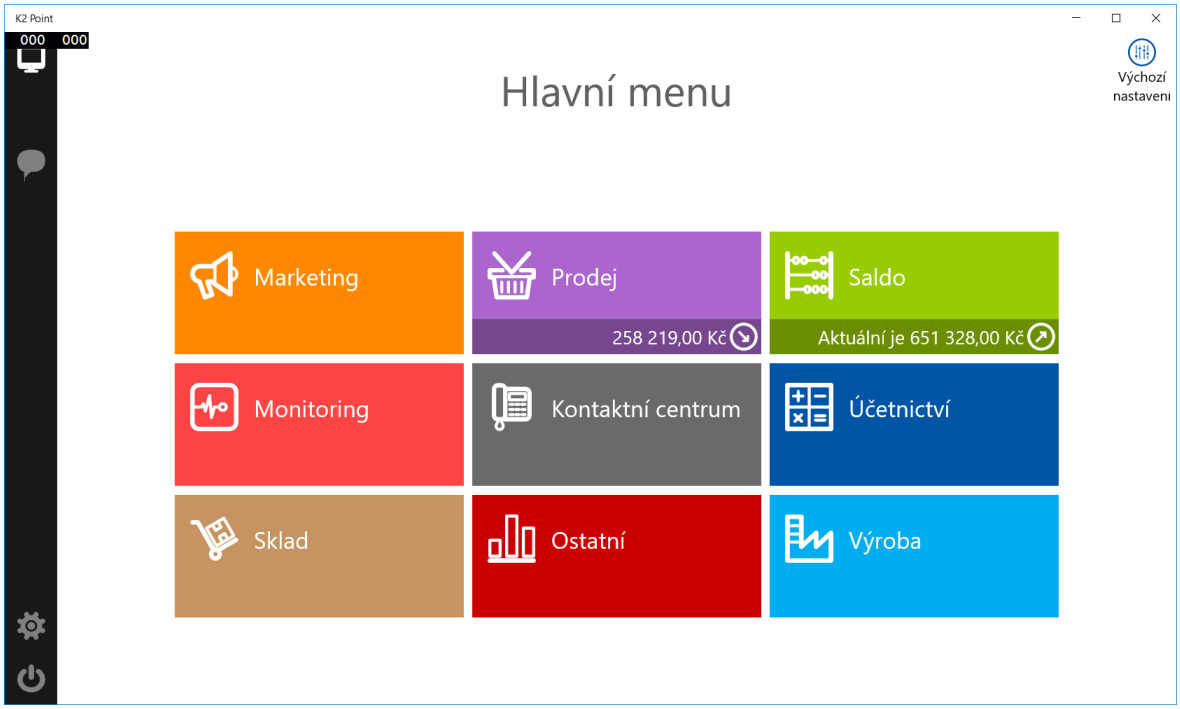
1 Úvod

Dashboard pro přehledné zobrazovaní dat firmy je program, který umožňuje na základě dat z webových služeb společnosti K2 atmitec dynamicky vytvářet menu, grafy a tabulky. Hlavním účelem programu je jednoduché a přehledné zobrazení dat. Program je určen pro počítače s operačním systémem Windows 10. Zobrazení je vytvořeno pomocí popisu zobrazení v datovém formátu JSON, které se stahuje z aplikačního rozhraní webových služeb. Data se následně zpracovávají a převádějí na prvky navigace, filtry dat a samotné zobrazovací prvky.

V části práce nazvané „Uživatelská dokumentace“ popisuji vzhled a funkci jednotlivých prvků z pohledu uživatele.

V další části nazvané „Programátorská dokumentace“ uvádím použité technologie a celkovou strukturu programu. Dále jsou zde uvedeny všechny důležité třídy projektu a popsány jejich vnitřní třídy, vlastnosti i metody.

2 Uživatelská dokumentace



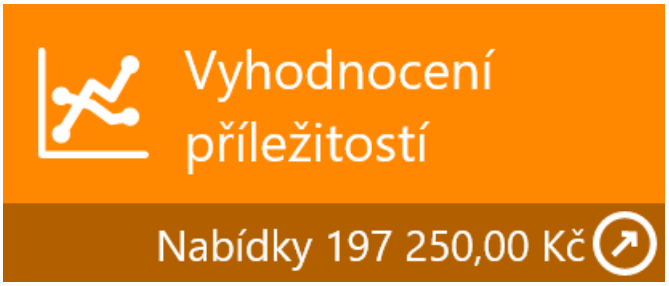
Obrázek 1 - Ukázka uživatelského rozhraní

2.1 Menu

Jednotlivé části programu jsou přístupné pomocí vertikálního navigačního menu, jež obsahuje čtyři tlačítka. Těmito tlačítky se dá přepínat mezi jednotlivými obrazovkami programu: hlavní obrazovka, obrazovka notifikací, obrazovka nastavení a poslední tlačítko slouží pro ukončení programu.

2.2 Hlavní obrazovka

Hlavní obrazovka je nejdůležitější součástí programu. Hlavní obrazovka se používá pro zobrazení navigačního menu a prezentaci dat v přehledné formě. Navigační menu je zobrazeno ve formě dlaždic pro jednoduchou navigaci k hledaným datům. Data jsou zobrazována ve formě různých grafů nebo přehledné tabulky.



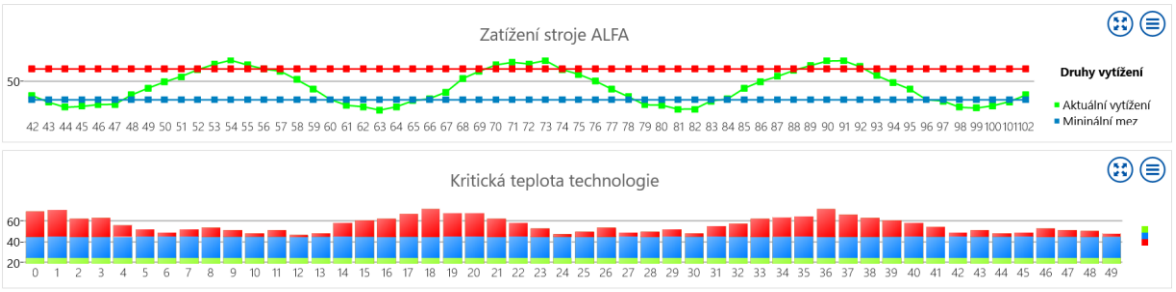
Obrázek 2 - Dlaždice navigačního menu

2.2.1 Navigační dlaždice

Dlaždice jsou nejdůležitější součástí navigačního menu, jež slouží pro přechod mezi skupinami dat a samotnými daty. Navigovat zpět v hierarchii menu lze pomocí tlačítka „Zpět“.

Dlaždice se skládají z ikony, názvu a některé dlaždice mají i informační panel, který zobrazuje vybraná důležitá data.

2.2.2 Grafy a jednoduché tabulky



Obrázek 3 - Ukázka zobrazení grafů

Pro zobrazování dat se používají speciální dlaždice, které obsahují název, samotný graf nebo tabulku, popis a tlačítka sloužící pro roztažení a změnu typu zobrazení dat. Typ zobrazení dat lze u každé dlaždice libovolně měnit. Každá dlaždice se dá roztáhnout na celou velikost okna pro zobrazení dat. Některé dlaždice fungují zároveň i jako filtry.

Podporované typy grafů: spojnicový, sloupcový, výsečový a plošný.

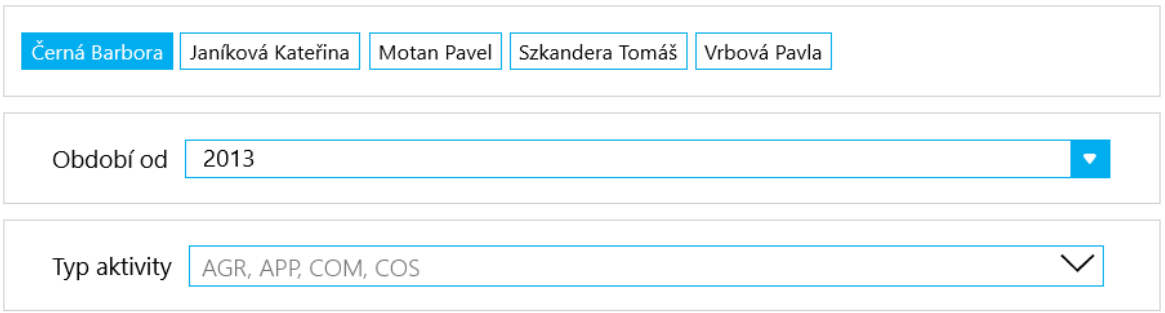
			- Všechna střediska		
				Ostrava Avion	Ostrava Futurum
			Obrat	Obrat	Obrat
+ Leden 2014		+ Všechny osoby	587 119 Kč	338 738 Kč	248 381 Kč
+ Únor 2014		- Všechny osoby	259 595 Kč	136 479 Kč	123 116 Kč
		Motan Pavel	259 595 Kč	136 479 Kč	123 116 Kč
+ Březen 2014		+ Všechny osoby	683 529 Kč	403 884 Kč	279 645 Kč
+ Duben 2014		+ Všechny osoby	706 041 Kč	418 248 Kč	287 793 Kč
+ Květen 2014		- Všechny osoby	827 868 Kč	514 875 Kč	312 993 Kč
		Motan Pavel	827 868 Kč	514 875 Kč	312 993 Kč
+ Červen 2014		- Všechny osoby	514 960 Kč	287 460 Kč	227 500 Kč
		Motan Pavel	514 960 Kč	287 460 Kč	227 500 Kč
+ Červenec 2014		+ Všechny osoby	581 499 Kč	313 938 Kč	267 561 Kč
+ Srpen 2014		+ Všechny osoby	653 130 Kč	363 899 Kč	289 231 Kč
+ Září 2014		+ Všechny osoby	718 679 Kč	407 009 Kč	311 670 Kč
+ Říjen 2014		+ Všechny osoby	787 777 Kč	430 299 Kč	357 478 Kč
- Listopad 2014		- Všechny osoby	801 018 Kč	466 130 Kč	334 888 Kč
		Motan Pavel	801 018 Kč	466 130 Kč	334 888 Kč
	05.11.2014	- Všechny osoby	801 018 Kč	466 130 Kč	334 888 Kč
		Motan Pavel	801 018 Kč	466 130 Kč	334 888 Kč
+ Leden 2015		+ Všechny osoby	613 059 Kč	351 069 Kč	261 990 Kč
+ Únor 2015		+ Všechny osoby	618 944 Kč	235 266 Kč	383 678 Kč
- Březen 2015		- Všechny osoby	671 503 Kč	385 003 Kč	286 500 Kč
		Motan Pavel	385 003 Kč	385 003 Kč	
		Vrbová Pavla	286 500 Kč		286 500 Kč
	13.03.2015	- Všechny osoby	385 003 Kč	385 003 Kč	
		Motan Pavel	385 003 Kč	385 003 Kč	
	14.03.2015	- Všechny osoby	286 500 Kč		286 500 Kč

Obrázek 4 - Ukázka kontingenční tabulky

2.2.3 Kontingenční tabulka

Tato speciální tabulka má za úkol přehledné zobrazení velkého množství dat. K tomu jí oproti jednoduché tabulce slouží nejenom barevné rozdělení buněk, ale i možnost otevírat a zavírat určité části tabulky.

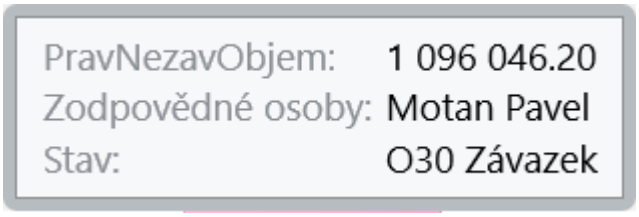
2.2.4 Filtry



Obrázek 5 - Filtry

Filtry slouží pro filtraci dat. Filtry se liší podle množství vybraných položek – jednopoložkové nebo vícepoložkové. Dále jsou filtry děleny na filtry s výčtem možností, filtry s rolovací nabídkou a filtry se stromem možností. Některé grafy a tabulky mohou sloužit i jako filtry, např. při výběru jedné výseče výsečového grafu. Filtry lze navrátit do výchozího nastavení pomocí tlačítka „Výchozí nastavení“.

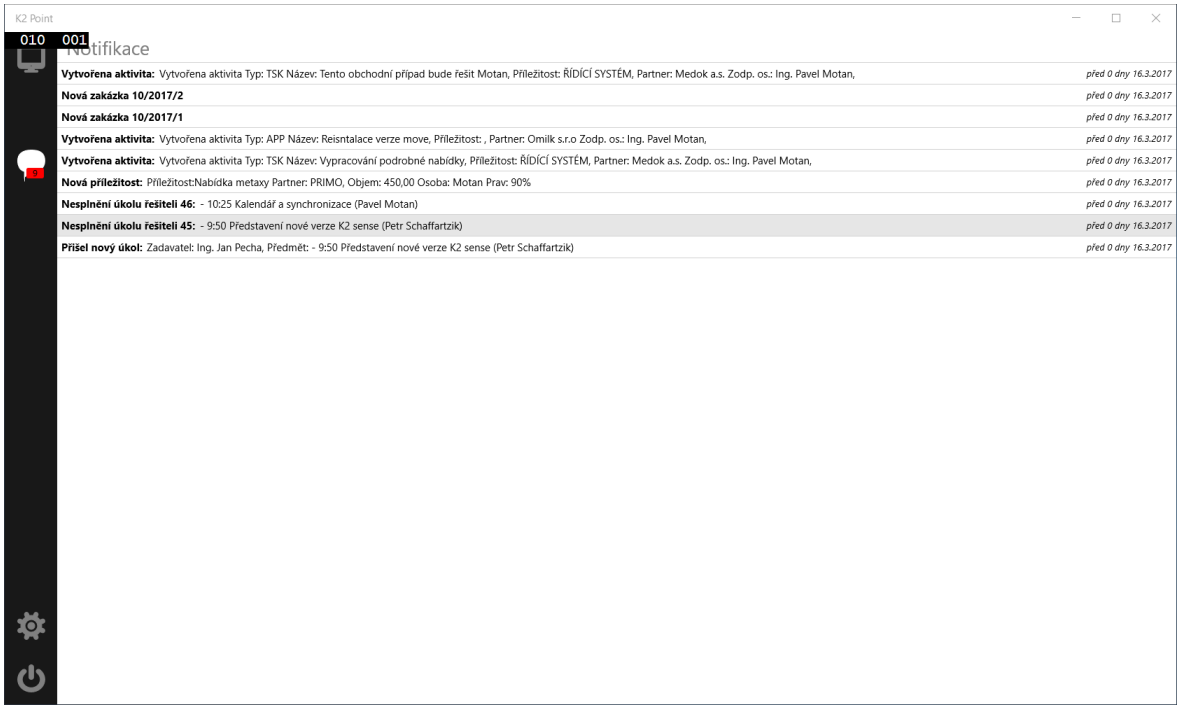
2.2.5 Popisky



Obrázek 6 - Popisek části grafu

Popisky slouží k zobrazení dodatečných informací o datech v grafu nebo tabulce nebo u některých tlačítek.

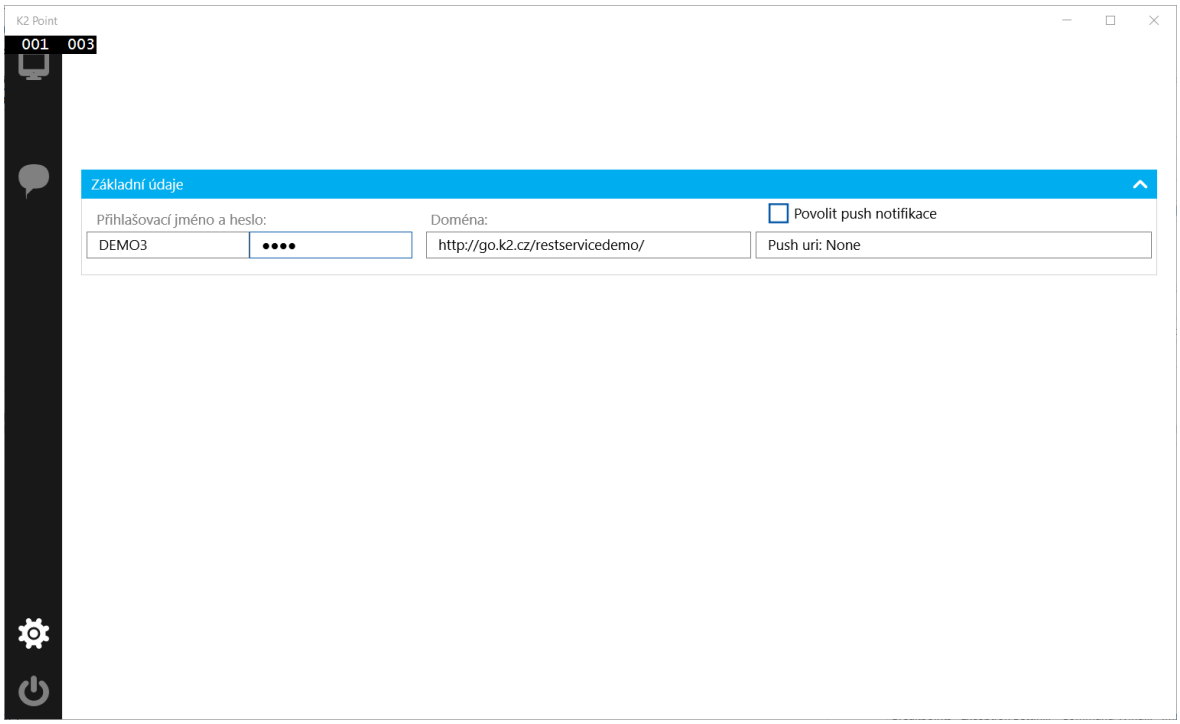
2.3 Obrazovka notifikací



Obrázek 7 - Obrazovka notifikací

Obrazovka notifikací slouží k zobrazení všech aktivních notifikací. Každá notifikace obsahuje zvýrazněný název notifikace, popis problému nebo zprávy a datum, kdy byla notifikace poprvé přijata. Každou notifikaci je možno deaktivovat kliknutím na ni. Zároveň se na ikoně obrazovky notifikací zobrazí celkový počet aktivních notifikací.

2.4 Obrazovka nastavení



Obrázek 8 - Obrazovka nastavení

Nejdůležitější součástí obrazovky nastavení jsou především textová pole pro zadání jména a hesla, která slouží k autorizaci pro server. Dále je v nastavení pole pro doménu, ze které bere program data. Nakonec je zde ještě checkbox povolující push notifikace pro Windows 10.

3 Programátorská dokumentace

3.1 Technologie

3.1.1 Universal Windows Platform

Universal Windows Platform je technologie společnosti Microsoft poprvé představená v operačním systému Windows 8. V jádru Universal Windows Platform aplikací je myšlenka, že uživatel chce mít aplikaci přenosnou mezi všemi zařízeními, podle toho, které je pro něj právě nejvhodnější.

Windows 10 zjednodušil vývoj aplikací pro Windows Universal Platform uvolněním jednotného API, jednoho balíčku a jednoho obchodu pro všechna zařízení využívající operační systém od společnosti Microsoft – počítač, tablet, chytrý telefon s Windows, Xbox, HoloLens, SurfaceView a další. Další zjednodušení plynou z podpory více velikostí obrazovek, a také podporou různých variací módů interakce, například dotyk, myš a klávesnice, herní ovladač nebo pero. [1]

3.1.2 Programovací jazyk C#

Jazyk C# je všestranně použitelný, typově bezpečný a objektově orientovaný programovací jazyk vyvinutý společností Microsoft. Jedním z nejdůležitějších hledisek při vývoji jazyka C# je produktivita. Samotný jazyk je nezávislý na platformě, ale většinou se využívá spolu s .NET Frameworkem.

Jazyk C# vychází z jazyka C++, ale v mnohém se od něj liší. Jazyk C# je objektově orientovaný jazyk, což znamená, že využívá konceptů uzavření, dědičnosti a polymorfismu. Jazyk C# dále přidává další možnosti psaní programů, jako jsou vlastnosti a události. Moderní podoba jazyka C# si ovšem vypůjčuje i určité koncepty z funkcionálních jazyků. Například v jazyce C# mohou být funkce pomocí delegátů použity i jako parametr jiné funkce. [2]

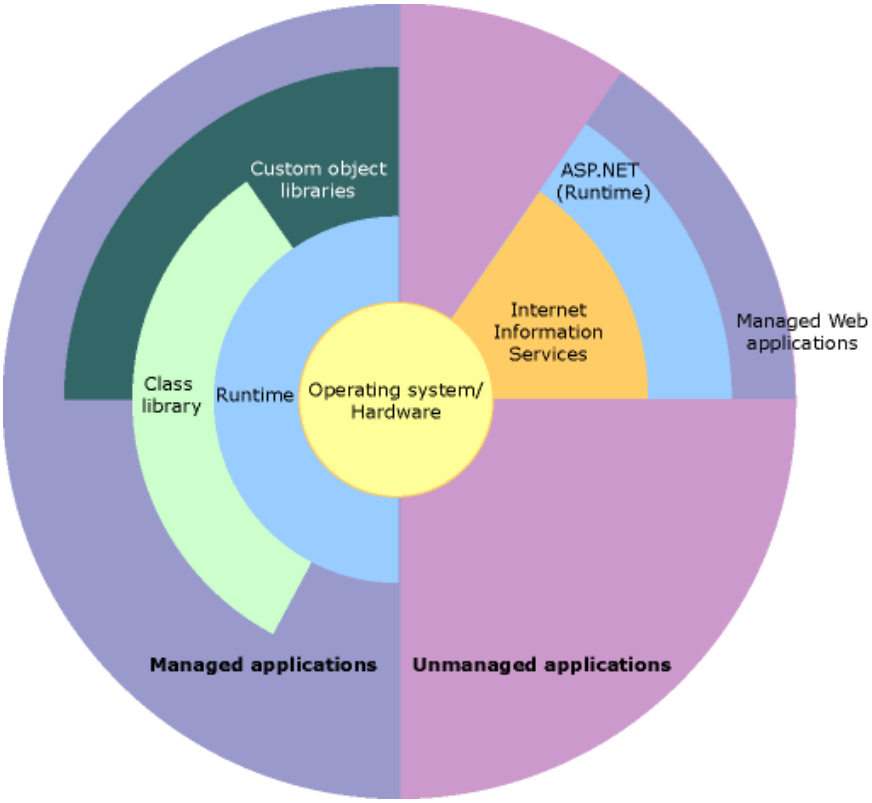
3.1.3 Jazyk Extensible Application Markup Language (XAML)

XAML je jazyk založený na XML vyvinutý společností Microsoft. XAML se používá pro vizuální reprezentaci aplikací založených na Universal Windows Platform a dalších. Pomocí jazyku XAML lze oddělit veškeré definice vzhledu od ostatního kódu. Toto oddělení umožňuje lepší spolupráci více lidí na jednom projektu. [3]

3.1.4 .NET Framework

.NET Framework se skládá z Common Language Runtime (CLR) a knihovny tříd .NET Frameworku. CLR provádí kód a dále se stará o správu paměti, správu vláken, ověřování zabezpečení kódu, kompilaci a další systémové funkce. Knihovna tříd je obsáhlá, objektově orientovaná kolekce knihoven,

která umožňuje vývoj aplikací od aplikací určených pro ovládání příkazovou řádkou, přes klasické desktopové aplikace využívající GUI, až po webové aplikace postavené na technologiích ASP.NET. [4]



Obrázek 9 - Ilustrace všech částí .NET Frameworku [3]

3.1.5 Knihovna WinRTXamlToolkit

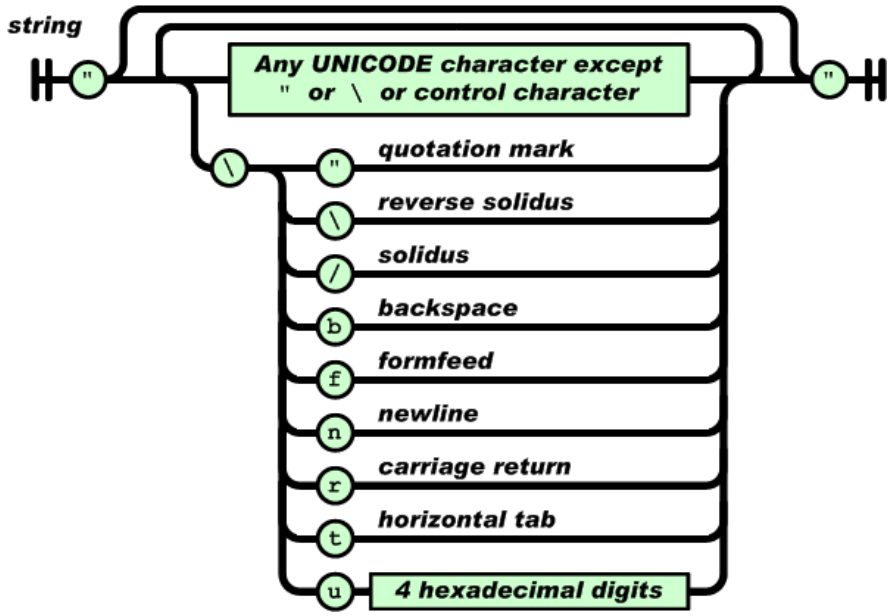
Tato knihovna je kolekce ovládacích prvků, nástrojů pro zobrazení dat, rozšíření a pomocných tříd. Součástí této knihovny jsou třídy pro vykreslování dat a uživatelských prvků, jako jsou grafy, tlačítka s vodoznakem, třída pro zobrazení stromové struktury apod. Dále obsahuje třídy pro převod různých typů dat a několik tříd pro pomoc s vyhledáním a opravou chyb. [5]

Já jsem v této práci využil především třídy pro grafy a třídu TreeView pro zobrazení stromové struktury. Třídu pro grafy jsem upravil z důvodu neexistence podpory přiřazení vlastní barvy k určité části grafu.

3.1.6 Datový formát JSON

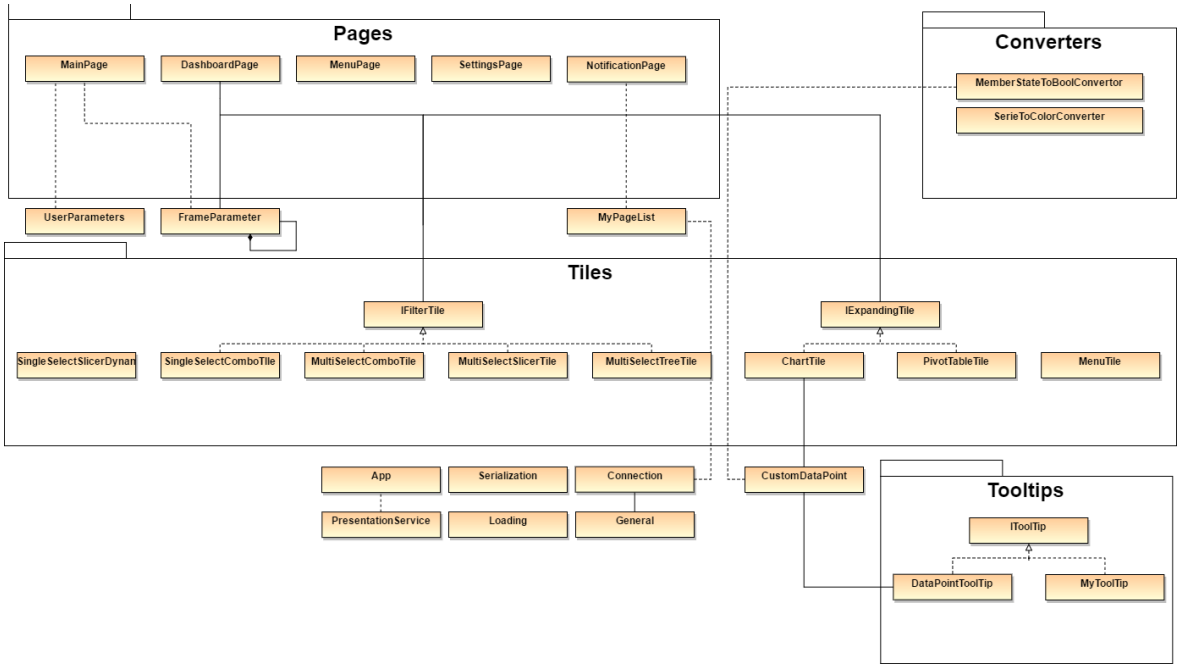
Datový formát JSON je jednoduchý formát pro výměnu dat čitelný člověkem. Dále je i jednoduchý pro počítače, jak pro generování, tak i rozložení. JSON je nezávislý na jakémkoli programovacím jazyku, ale následuje konvence udané jazyky C, C++, C#, Java a podobnými. Tyto vlastnosti dělají z JSON formát ideální pro výměnu dat mezi různými systémy. JSON může mít dvojí podobu. První podoba je seznam párů jmen a hodnot. Další možností je zápis v podobě seřazeného seznamu hodnot. [6]

V mém projektu používám formát JSON pro komunikaci s API.



Obrázek 10 - Příklad zápisu hodnoty řetězce ve formátu JSON [5]

3.2 Struktura programu



Obrázek 11 - Struktura programu v UML

Program se skládá z různých skupin tříd, kde každá skupina řeší podobné problémy. Skupina Pages řeší vykreslování okna programu. Zde je specifická třída MainPage, která se stará o postranní menu. Skupina Converters konvertuje určitá data mezi typy potřebnými pro výpočet a typy používanými prvky uživatelského rozhraní. Skupina Tooltips slouží k zobrazení dodatečných dat o prvcích na obrazovce. Nejpočetnější skupina Tiles, slouží k samotnému dynamickému zobrazení veškerých dat příchozích z API. Tato skupina obsahuje navigační dlaždici, dlaždice pro zobrazení dat a nakonec všechny používané filtry. Ostatní třídy slouží každá ke svému specifickému účelu, jež je popsán níže.

3.3 Popis důležitých tříd programu

3.3.1 Třída App

Hlavní třída programu. Objekt ze třídy App se vytvoří na začátku programu a stará se o základní nastavení programu a zavedení potřebných služeb jako PresentationService. Dále zpracovává změnu životního cyklu aplikace.

a. Metoda OnLaunched
   Nejdůležitější metodou třídy App je metoda OnLaunched, která se volá hned po spuštění programu a jejím úkolem je nastavit výchozí obrazovku programu. Pokud je v nastavení uloženo povolení push notifikací metoda OnLaunched automaticky obnoví notifikační kanál.

b. Metoda GetCulture
   Pomocná metoda GetCulture slouží pouze k získání výchozího jazyku, který je jako první uveden v systému Windows. Následně informaci uloží, aby se textové zdroje mohly patřičně upravit.

c. Metoda GetUserId
   Pomocná metoda GetUserId je první metoda, která využívá aplikačního rozhraní společnosti K2, aby získala informace o aktuální kontaktní osobě, která právě tyto služby využívá. Nejdůležitější získanou informací je samotné id kontaktní osoby, které se poté využívá pro získání kanálu pro push notifikace.

2.3.2 Třída MainPage

Zobrazení třídy MainPage je rozděleno mezi dvě základní části: navigační panel a Frame (třída umožňující zobrazení jiné obrazovky). Úlohou třídy MainPage je navigace mezi obrazovkami programu a správné zobrazování a reakce navigačního panelu. K tomu třídě slouží několik metod reagujících na události způsobené kliknutím na tlačítko navigačního panelu. V případě, že si uživatel zvolí hlavní obrazovku s dashboardy třída MainPage zjistí id výchozího layoutu a pošle ho jako parametr třídě DashboardPage.

a. Metoda ImagePlaceholder_Loading
   Metoda ImagePlaceholder_Loading reaguje na událost způsobenou potřebou programu načíst grafické zdroje pro zobrazení jednotlivých prvků navigačního panelu.

b. Metoda ImagePlaceholder_Tapped
   Tato metoda je nejdůležitější metodou navigačního panelu, neboť reaguje na událost způsobenou kliknutím na některé z tlačítek určených pro navigaci.

c. Metoda GreyAllButtons
Tato metoda zvýrazní tlačítko s obrázkem reprezentujícím právě zvolenou obrazovku. Jediný parametr určuje název zvýrazněného tlačítka. Metoda funguje tak, že nejprve „znevýrazní“ všechna přítomná tlačítka a poté zvýrazní pouze určené, čímž odpadá nutnost si pamatovat naposledy zvýrazněné tlačítko.

```
/// <summary>
/// Grays all buttons except one
/// </summary>
/// <param name="whiteButton">White button</param>
public void GreyAllButtons(string whiteButton)
{
    SolidColorBrush scb = new SolidColorBrush(Colors.Gray);
    SolidColorBrush white = new SolidColorBrush(Colors.White);

    // reset color of all buttons
    ButtonPaths.ForEach(bp =>
    {
        bp.Stroke = scb;
        bp.Fill = scb;
    });

    // higlight current button
    Windows.UI.Xaml.Shapes.Path whitePath = ButtonPaths.FirstOrDefault(bp => bp.Name == whiteButton);
    if (whitePath != null)
    {
        whitePath.Stroke = white;
        whitePath.Fill = white;
    }
}
```

Obrázek 12 - Metoda GreyAllButtons

d. Metoda GetDefaultPage
Tato metoda volá API, kvůli potřebě získat uživatelské parametry, jímž je i id výchozího layoutu.

3.3.3 Třída DashboardPage

Tato třída slouží ke stahování a zobrazování dynamických layoutů z API. Zobrazuje jak dynamická navigační menu, tak samotné obrazovky zobrazující filtry a data. Tato třída je tím pádem jedna z nejdůležitějších v celém programu, protože převádí layout z API na reálné rozvržení dlaždic.

a. Metoda OnNavigatedTo
Tato metoda reaguje na událost způsobenou změnou vnitřní obrazovky třídy MainPage. Jejím úkolem je zkontrolovat správnost parametru a uložit data. Poté volá metodu Page_Download.

```
Parameter = e.Parameter as FrameParameter;
if (Parameter != null && Parameter.NextPage != "")
{
    if(SettingsPage.settingsChanged)
    {
        App.GetUserId();
        SettingsPage.settingsChanged = false;

        long? defaultPage = await MainPage.GetDefaultPage();
        MainPage.DashboardDefaultPage = defaultPage ?? 0;
        MainPage.MainPageReference.NavigateDashboard(MainPage.DashboardDefaultPage.ToString());
        return;
    }

    Page_Download();

    if (Parameter.BackParameter == null)
        BackButton.Visibility = Visibility.Collapsed;
}
```

Obrázek 13 - Ukázka kódu metody OnNavigatedTo

b. Metoda Page_Download
Metoda Page_Download je volána následně po kontrole navigačního parametru. Jejím úkolem je stažení layoutu označeného v parametru jako aktuální a kontrola platnosti přijatých dat. Pokud dojde k chybě, má za úkol na základě odpovědi serveru rozpoznat, zda se jedná o chybnou doménu, přihlašovací údaje či špatné identifikační číslo layoutu. Pokud k chybám nedojde stáhne další údaje obsahující meta-data o daném layoutu. Z meta-dat nás zajímá především titulek layoutu. Všechna data, která jsou v průběhu metody stažena, jsou ukládána do paměti. V poslední fázi metody se na základě typu stažených objektů zavolají metody pro vykreslení navigačního menu nebo obrazovky s daty.

c. Metoda GetTiles
Tato metoda je volána na konci metody Page_Download pokud je typem staženého prvku třída s daty pro dlaždici navigačního menu. Na počátku běhu má tato metoda za úkol na základě popisu v datech layoutu připravit kontejnery obsahující cílové dlaždice pro horizontální nebo vertikální orientaci. Poté se iterují další části layoutu a u každé se určuje směr kontejneru a následně se iterují všechna data o dlaždicích danému kontejneru náležející. Každé dlaždici jsou přiřazena data z layoutu a je přidána obsluha události kliku na dlaždici. Obsluhující funkce vytvoří nový navigační parametr s identifikačním číslem příštího layoutu a starý parametr uloží do proměnné, aby se dalo v navigačním menu vracet zpět a poté naviguje na DashboardPage s novým parametrem. V samotném závěru metody se ověří, zda menu nepřesáhlo limity okna. Pokud ano, vytvoří lištu pro skrolování.

d. Metoda GetCharts
Tato metoda je volána na konci metody Page_Download pokud je typem staženého prvku třída s daty pro grafy nebo filtry. Jako první předpočítá na základě orientace kontejnerů a minimálních velikostí jednotlivých dlaždic minimální horizontální a vertikální velikost celého zobrazení. Poté tyto velikosti porovná s aktuální velikostí okna zmenšenou o velikost levého panelu. Pokud zobrazení přesahuje jeden nebo oba parametry jsou vygenerovány lišty pro skrolování a bude se s přesahem počítat u generování filtrů a grafů. Poté se přistupuje

k samotnému generování kontejnerů a ostatních elementů. Na základě typu dat jsou vytvářeny jednotlivé typy elementů. Navíc se ještě objevuje speciální případ kontejneru s filtry, který říká, že filtry musí být nad všemi ostatními kontejnery a pokud tento není ve výčtu kontejnerů jako první musí být tento případ ošetřen.

e. Metoda BackButton_Tapped
Tato metoda reaguje na událost způsobenou kliknutím na tlačítko „Zpět“. Jediným úkolem metody je vytáhnout z navigačního parametru parametr předchozí a navigovat s ním na stránku DashboardPage.

f. Metoda DefaultButton_Tapped
Tato metoda reaguje na událost způsobenou kliknutím na tlačítko „Výchozí nastavení“. Metoda, pokud je na layoutu se zobrazením filtrů a dat, zavolá API k resetu všech filtrů, vyresetuje použité kontejnery a další objekty pro zobrazení a z odpovědi serveru zobrazí data ve výchozím nastavení.

```
/// <summary>
/// Resets filters and redraws
/// </summary>
/// <param name="sender">Button</param>
/// <param name="e">Args</param>
private async void DefaultButton_Tapped(object sender, TappedRoutedEventArgs e)
{
    if (Charts)
    {
        Load.IsVisible = true;
        string result = await Connection.GetDataAsync("Data/ARDashboard/ResetFilters/" + Parameter.NextPage);
        CurrentDashboardData = Serialization.GetObjFromJSON<ARDashboardData>(result);

        MainFrame.Children.Clear();
        FilterFrame.Children.Clear();
        StackPanels.Clear();

        GetCharts(CurrentDashboardData);
        Load.IsVisible = false;
    }
}
```

Obrázek 14 - Ukázka kódu metody DefaultButton_Tapped

g. Metoda GetRightTypeOfFilter
Metoda odděluje výběr správného typu dlaždice filtru podle dat z funkce GetCharts.

h. Metoda GetDataForChange
Tato metoda nejprve uloží stav filtru na server. Poté postupně prochází všechny závislé části. Z každé závislé části vezme data a pošle je na API. Z API následně přijde odpověď s novými daty, které se použijí pro vykreslení grafů odpovídajících novým filtrům.

i. Metoda ChangeTile
Metoda ChangeTile prohledá všechny kontejnery a pomocí jednoduché lambda funkce v nich vyhledává cílový graf nebo tabulku, které se mění data. Data se mění nejčastěji z důvodu změny některého z filtrů. Pokud svůj cíl najde, tak nahradí jeho data novými a zavolá metodu pro překreslení.

j. Metoda EnlargeTile
Tato metoda zvětší vybranou dlaždici na celou obrazovku určenou pro zobrazení data a zruší viditelnost ostatních kontejnerů a dlaždic.

k. Metoda ReturnSizes
Tato metoda pouze ruší změny způsobené metodou EnlargeTile.

3.3.4 Třída NotificationPage

Úkolem třídy NotificationPage je zobrazení všech aktivních notifikací přihlášeného uživatele. Notifikace se zobrazují ve vertikálním seznamu. Každá notifikace se skládá z názvu, popisu, data vytvoření a GUID, které se nezobrazuje, ale slouží jako identifikátor pro deaktivování notifikací.

a. Vnitřní třída MyListViewItem
Tato třída slouží pouze pro uložení všech údajů o notifikaci. Třída se používá jako jeden prvek pro objekt třídy ListView, který se stará o zobrazení seznamu notifikací.

b. Metoda GetNotification
Tato metoda volá API ke stažení údajů o všech aktivních notifikacích a následně volá metodu ShowInformation.

```
<ListView.ItemTemplate>
    <DataTemplate>
        <Grid>
            <Grid.ColumnDefinitions>
                <ColumnDefinition Width="*" />
                <ColumnDefinition Width="Auto" />
            </Grid.ColumnDefinitions>
            <StackPanel Grid.Column="0" Orientation="Horizontal" HorizontalAlignment="Left">
                <TextBlock Text="{Binding Path=Caption}" FontWeight="Bold" FontSize="12" />
                <TextBlock Text="{Binding Path=Text}" Margin="5,0,0,0" FontSize="12" />
            </StackPanel>
            <StackPanel Grid.Column="1" Orientation="Horizontal" HorizontalAlignment="Right">
                <TextBlock Text="{Binding Path=Date}" FontStyle="Italic" FontSize="10.667" />
            </StackPanel>
        </Grid>
    </DataTemplate>
</ListView.ItemTemplate>
```

Obrázek 15 - Ukázka kódu pro vzhled položky notifikace v XAML

c. Metoda ShowInformation
Tato metoda rozparsuje data stažená z API do jednotlivých instancí třídy MyListViewItem a uloží je do struktury, která má na starost zobrazení seznamu notifikací: instance třídy ListView.

d. Metoda listview_SelectionChanged
Tato metoda zachycuje událost, která je aktivována při kliknutí na notifikaci. Metoda zjistí GUID dané notifikace, kterou pomocí API deaktivuje a voláním metody GetNotification obnoví seznam notifikací.

3.3.5 Třída SettingsPage

Tato třída slouží k nastavení autorizačních údajů, názvu webového serveru a dále obsahuje pole pro povolení a zobrazení URI kanálu pro push notifikace. Autorizační údaje se skládají z přihlašovacího jména a hesla. Webový server se určuje pomocí URL, na které běží API společnosti K2. Údaje se ukládají automaticky při změně, není vyžadováno potvrzení pomocí tlačítka.

a. Metoda PushNotificationBox_Tapped
   Tato metoda reaguje na událost vyvolanou stiskem checkboxu pro povolení push notifikací. V případě povolení push notifikací je pomocí statické třídy Connection vytvořen kanál pro push notifikace a jeho URI je poté vypsána v poli pro URI kanálu. V případě zákazu je kanál push notifikací uzavřen.

b. Metody LoginTextBox_TextChanged, DomainTextBox_TextChanged, PasswordBox_PasswordChanged
   Metody jsou volány vždy, když dojde ke změně v jejich specifickém textovém poli a automaticky změny ukládají.

c. Metoda NotificationUriBox_TextChanged
   Metoda je specifická tím, že jejím jediným úkolem je hlídat nepřepsání obsahu textového pole pro URI kanálu push notifikací. Proto vždy při změně obsahu textového pole, je volána tato metoda a vrátí obsah do původní formy. Textové pole není deaktivováno z důvodu umožnění jednoduchého zkopírování URI do schránky.

3.3.6 Třída DataPointToolTip

První ze tříd pro popisky se jmenuje DataPointToolTip a používá se k zobrazení dat o částech grafu, jako jsou sloupce, body na spojnicovém grafu nebo výseč koláčového grafu. Pro získání dat využívá třídy CustomDataPoint, která ukládá informaci o jedné části grafu.

a. Statická metoda DataChanged
   Tato statická metoda reaguje na vyvolání události způsobené změnou dat daného popisku. Úkolem je obnovit načtení a zobrazení nových dat.

b. Metoda GetToolTipData
   Metoda GetToolTipData je nejdůležitější metodou třídy DataPointToolTip. Jejím úkolem je získat z objektu třídy CustomDataPoint všechna potřebná data k zobrazení popisku. V objektu jsou mimo jiné uložena surová data z API, podle kterých se zjišťují názvy a hodnoty parametrů dané části grafu.

c. Metoda GetRow
   Tato metoda získá již konkrétní data pro jeden řádek popisku. Metoda mimo jiné využívá i regex, pro zjištění specifické struktury položky. Vyparsovaná data následně zašle jako parametry při volání metody AddRowToOutput.

d. Metoda AddRowToOutput
   Tato poslední metoda třídy DataPointTooltip v posloupnosti metod, jež řeší získání a vykreslení dat má za úkol vytvořit layout popisku. Layout se tvoří dynamicky, pro každý řádek se dynamicky vytvoří objekty TextBlock a Grid, což jsou nativní UI objekty. Každé zavolání

této metody vytvoří pouze jeden řádek celkového layoutu, který se poté přidává do seznamu řádků. Seznam řádků se následně zobrazí.

3.3.7 Třída MyToolTip

Třída MyToolTip slouží k zobrazení popisku vysvětlujícího funkci tlačítek zobrazených v hlavní obrazovce. Konkrétně se jedná o tlačítka „Zpět“ a „Výchozí nastavení“. Popisek obsahuje obrázek tlačítka, název a popis funkce.

a. Setter vlastnosti ImageIdentifier
   Tato vlastnost, při přiřazení hodnoty řetězce vygeneruje potřebný obrázek.

3.3.8 Třída ChartTile

Úkolem této třídy je zobrazovat a obsluhovat různé typy grafů a tabulku. Tato třída se stará o správné rozparsování dat, převod do různých typů grafů, ale i reakci na klik, pokud je graf zároveň i filtr. Další možností třídy je změna grafu na přehlednou kontingenční tabulku, která přehledně zobrazí dostupná data.

a. Vnitřní třída SerieTag
   Jediným úkolem třídy SerieTag je udržení dat. Tato třída se poté využije pro vnitřní data buněk kontingenční tabulky.

b. Metoda ReturnToDefault
   Tato metoda pouze vyresetuje současná nastavení grafu a ukáže původní zobrazení. Je volána při vybrání možnosti „Výchozí nastavení zobrazení“ v menu grafu.

c. Metoda DrawChart
   Tato metoda se volá vždy při potřebě vykreslit graf nebo tabulku. Jejím úkolem je mimo jiné zapnout animaci načítání pro uživatele, aby věděl, že data jsou zpracovávána. Dále má ještě jednu, důležitější funkci a to rozhodnout, kterou vykreslovací metodu použít na základě vybraného typu grafu nebo tabulky. Poté asynchronně zavolá danou metodu s nízkou prioritou, aby nedocházelo k zasekávání vlákna pro uživatelské rozhraní.

d. Metoda ChooseGraphType_Tapped
   Při kliku na menu grafu je vyvolána událost, která je zpracována právě touto metodou. Jejím úkolem je nastavit potřebné příznaky patřící požadovanému typu grafu nebo tabulky. Poté volá výše popsanou metodu DrawChart, která se postará o vykreslení požadovaných změn.

e. Metoda SetChart

Metoda SetChart již slouží k vytvoření samotného grafu z dat pro jeho následné vykreslení. V počátku uvede současné zobrazovací pole a data do výchozího stavu. Poté, pokud má graf více částí rozdělí je na jednotlivé části – například, pokud graf zobrazuje více spojnicových grafů na jedné plošině. U každé části je poté zjištěn její typ a podle toho jsou grafu přiřazeny styly, přizpůsobené každému typu grafu. V případě, že graf slouží také jako filtr jsou nastaveny potřebné značky a metody obsluhující událost kliku na graf. Nakonec jsou části grafu přiřazena data a je připraven být vykreslen. K tomu je uložen do seznamu již sestavených částí a při dokončení metody je graf vykreslen.

```
// Filters
if (Data.Dependents != null && Data.Dependents.Count > 0)
{
    dps.IsSelectionEnabled = true;
    dps.Tag = serie;
    dps.Tapped += Filter_Tapped;
}

// Data
AllPointsInSerie = new List<CustomDataPoint>();
for (int i = 0; i < serie.DataPoints.Count; i++)
{
    string color = CurrentType == ARPresentationType.ptPieChart ? Palette[i % Palette.Length] : serie.Color;
    if (color == "")
        color = Palette[0];

    AllPointsInSerie.Add(new CustomDataPoint(serie.DataPoints[i], color, dps,
        Data.AxesDefinition, Data.QueryDefinition.DynamicQuery.LinesAndColumns));
}

dps.ItemsSource = AllPointsInSerie;
MainChart.Series.Add(dps);
```

Obrázek 16 - Ukázka kódu z metody SetChart

f. Metoda SetStackedChart

Tato metoda je volána pouze chceme-li vykreslit skládaný sloupcový graf. Vzhledem ke složitosti grafu a použitému frameworku je k vykreslení skládaného sloupcového grafu využito instancí jiných tříd. Metoda také v počátku uvede data do výchozího stavu. Poté zajistí všechny potřebné značky a metody ke správné funkci filtrů. Nakonec jsou grafu přiřazeny styly a všechna data, jako názvy jednotlivých částí grafu v legendě. Výsledný graf je přiřazen do seznamu grafů a zobrazen.

g. Metoda AddStackedChartToSeries

Tato metoda je velmi podobná metodě předchozí, ale jejím úkolem je vytvořit speciální případ skládaného sloupcového grafu, který je přidán k ostatním částem grafu a bude tak zobrazen spolu s dalšími částmi na jedné ploše.

h. Metoda SetGrid
Pomocí metody SetGrid jsou vypisovány data do dynamicky vygenerované tabulky. Vzhledem k neexistenci složitých datových tabulek ve frameworku Univesal Windows Platform jsou veškeré tabulky dynamicky generovány z ostatních grafických prvků. Hlavním prvkem, který určuje celkovou strukturu a udržuje ostatní grafické bloky v potřebných pozicích, je objekt třídy Grid. Na základě počtu dat je spočítána potřebná velikost tabulky a instanci třídy Grid jsou nastaveny parametry pro určení velikosti řádku a sloupce. Poté jsou jako vnitřní objekty tabulky použity objekty třídy Rectangle se specifickými styly, které jsou zodpovědné za vzhled tabulky a také za směr zarovnání textu. V průběhu se využívá metody GetGridSizes, která kromě potřebné velikosti vrací i všechna data již v objektu třídy UIElement – tudíž jsou připraveny k zobrazení.

i. Metoda Dps_LegendItemCreated
Jediným úkolem této jednoduché metody je zobrazení barvy grafu patřící k dané legendě.

j. Metoda GetGridSizes
Metoda GetGridSizes zjišťuje potřebnou šířku buňky tabulky tak, že ji zabalí do grafických objektů třídy TextBlock, které se přidají všechny potřebné části, které chceme zobrazit – tedy data třídy Border a data pro nastavení pozice v objektu třídy Grid. Poté, když jsou nastaveny styly všech grafických objektů a okrajů a nastavena data pro vykreslení, je zjištěna potřebná velikost buňky. Již vygenerované buňky se ukládají a na konci funkce se vrací spolu s výstupní hodnotou, abychom již nemuseli objekty generovat znovu. Takovýmto způsobem jsou vygenerovány všechny buňky a jejich velikosti jsou poté porovnány a pro každý sloupec je určena největší potřebná šířka, která se poté použije na všechny buňky sloupce. Stejně jako u grafů i tabulky mohou být zároveň filtry, a proto jsou při generování nastavována data a obsluha událostí pro kliknutí na buňku.

k. Metoda CreateDefaultTextBlock
Toto je pomocná metoda, která vytvoří výchozí TextBlock používaný pro zobrazení jednotlivých buněk. Třída element dynamicky vytvoří a nastaví všechny potřebné výchozí parametry jako je vzdálenost textu od okrajů buňky. Výsledný element je poté vrácen k dalšímu použití.

```
/// <summary>
/// Creates textblock and measures it
/// </summary>
/// <param name="Title">Text of textblock</param>
/// <param name="GridRow">Row index</param>
/// <param name="GridColumn">Column index</param>
/// <returns>Measured textblock</returns>
private TextBlock CreateDefaultTextBlock(string Title, int? GridRow = null, int? GridColumn = null)
{
    TextBlock output = new TextBlock();
    output.Text = Title;
    if (output.Text == string.Empty)
        output.Text = "0";
    //output.Padding = new Thickness(20, 0, 5, 0);
    output.Padding = new Thickness(10, 1, 10, 1);
    output.Measure(Windows.Foundation.Size.Empty);

    if(GridRow != null)
        Grid.SetRow(output, GridRow ?? 0);

    if(GridColumn != null)
    Grid.SetColumn(output, GridColumn ?? 0);

    output.HorizontalAlignment = HorizontalAlignment.Right;
    output.VerticalAlignment = VerticalAlignment.Center;
    return output;
}
```

Obrázek 17 - Ukázka kódu metody CreateDefaultTextBlock

l. Metoda Filter_Tapped
Tato metoda obsluhuje událost vyvolanou kliknutím na některý z grafových nebo tabulkových filtrů. V závislosti na typu prvního parametru, tedy odesílatele události, se z daného objektu vytáhnou všechna data potřebná pro filtr a zavolá se funkce FilterCall.

m. Metoda FilterCall
V metodě FilterCall jsou data získaná v parametru metody rozdělena na jednotlivé závislé části, které byly aktivovány kliknutím. Každá část má svůj identifikátor a data s ní související. Takto získaná data jsou poté odeslána na API. Při úspěšné odpovědi ze serveru je třída DashboardPage, která spravuje všechny zobrazené dlaždice notifikována o změně data potřebě překreslit obraz.

n. Metoda GetMembers
Tato metoda vrací všechna data, která souvisí s vybraným bodem grafu nebo buňkou tabulky, osami a závislými částmi. Všechna získaná data jsou uložena de seznamu a vrácena.

o. Metody CheckAxis
Tyto metody jsou pomocné pro metodu GetMembers. Slouží pro získání dat z jednotlivých os grafu. První z těchto metod bere jako parametry informace o grafu, první ose a závislých částech. Druhá bere jako parametry data z elementu, podle kterého chceme filtrovat, údaje o druhé ose a údaje o závislých částech. Obě tyto metody jsou nutné pro správnou funkci

filtrování, protože podle zjištěných identifikátorů a dat může server, na kterém běží API zjistit, jaký filtr je požadován a poslat žádaná data.

p. Metoda Expand
Tato metoda je volána, pokud uživatel chce roztáhnout dlaždici grafu na celou obrazovku určenou pro zobrazení grafů.

q. Metoda Compress
Metoda Compress je v podstatě opakem výše popsané metody Expand. Pokud již uživatel nechce mít dlaždici roztaženou zavolá se tato metoda, která vrátí dlaždici do původního stavu.

r. Metoda ChangeExpandButtonImage
Úkolem této metody je změna zobrazení tlačítka zvětšení nebo zmenšení dlaždice. Metoda vždy odstraní data pro vykreslení předchozího obrázku a vygeneruje nový.

3.3.9 Třída MenuTile

Třída MenuTile je dlaždice, která je součástí dynamického menu vygenerovaného z dat z API. Celá dlaždice se skládá z obrázku, názvu, specifické barvy a někdy také informačního panelu. Seznam těchto dlaždic slouží k navigaci mezi jednotlivými daty, o které máme zájem.

a. Settery vlastností ImagePath a LayoutImagePath
Tyto vlastnosti ukládají řetězec identifikující chtěný obrázek pro dlaždici. Tyto vlastnosti navíc automaticky načítají potřebné obrázky a přiřazují je zástupným grafickým elementům bez dat.

b. Konstruktor
V kontruktoru třídy MenuTile se získaná data rozdělí na jednotlivé elementy a výsledná dlaždice je okamžitě připravena k zobrazení.

c. Metoda GetColorFromString
Tato metoda slouží k převedení barvy ve formátu šesti hexadecimálních čísel do decimální reprezentace a poté na objekt třídy SolidColorBrush, který se ve frameworku UWP používá k nastavení barvy.

```
/// <summary>
/// Creates Brush for tile from string
/// </summary>
/// <param name="ColorString">Color code</param>
/// <param name="dark">If color should be darkened</param>
/// <returns>Brush with defined color</returns>
private Brush GetColorFromString(string ColorString, bool dark = false)
{
    if (ColorString == string.Empty)
        return new SolidColorBrush(Colors.Black);
    Color col = new Color();
    string R = (ColorString[0].ToString() + ColorString[1].ToString());
    string G = (ColorString[2].ToString() + ColorString[3].ToString());
    string B = (ColorString[4].ToString() + ColorString[5].ToString());
    byte Rb = (byte)General.ParseHexadec(R);
    byte Gb = (byte)General.ParseHexadec(G);
    byte Bb = (byte)General.ParseHexadec(B);
    col.A = 255;
    if(dark)
    {
        Rb = (byte)(Rb * 0.7);
        Gb = (byte)(Gb * 0.7);
        Bb = (byte)(Bb * 0.7);
    }

    col.R = Rb;
    col.G = Gb;
    col.B = Bb;
    return new SolidColorBrush(col);
}
```

Obrázek 18 - Ukázka kódu metody GetColorFromString

3.3.10 Rozhraní IFilterTile

Toto rozhraní slouží pro škálu různých filtrů, které nejsou součástí grafu nebo tabulky.

a. Metoda GetData
Tato metoda slouží k získání všech aktuálně vybraných dat, kterými se určuje, jaká data chceme zobrazit.

b. Metoda GetMembers
Tato metoda slouží k získání všech součástí filtru. Seznam částí je důležitý pro odeslání na API, když chceme filtrovat data.

3.3.11 Třídy filtrů

a. Třída SingleSelectSlicerDynamicTile
   Tento filtr je specifický, protože se může dynamicky měnit na základě dat z API. Funguje podobně jako MultiSelectSlicerTile popsaný níže, ale může mít vybraný pouze jeden prvek.

```
/// <summary>
/// When tapped on button
/// </summary>
/// <param name="sender">Button</param>
/// <param name="e">Args</param>
private async void Toggle_Tapped(object sender, TappedRoutedEventArgs e)
{
    foreach (ToggleButton toggle in MainStack.Children)
        toggle.IsChecked = false;

    (sender as ToggleButton).IsChecked = true;
    int tagId = (sender as ToggleButton).Tag as int? ?? 0;

    string result = await Connection.GetDataAsync(
        string.Format("Data/ARDashboard/DFEvaluateFilter/{0}/{1}", Data.ID, tagId));
    ARDynFilterEvalResult dynRes = Serialization.GetObjFromJSON<ARDynFilterEvalResult>(result);

    ToolTipService.SetToolTip(Info, dynRes.TooltipText);
    if (DashboardPage.PageContainer != null)
        DashboardPage.PageContainer.GetDataForChange(Data, dynRes.Members.Members);
}
```

Obrázek 19 - Ukázka metody volané při změně filtru

b. Třída SingleSelectComboTile
   Hlavní dominantou tohoto filtru je ComboBox, který umožňuje vybrat jednu položku ze seznamu dat.

c. Třída MultiSelectComboTile
   Tato třída je na první pohled podobná třídě SingleSelectComboTile. Hlavní součástí je opět ComboBox, ale každý prvek obsahuje název checkbox. Pomocí checkboxů je možno vybrat více prvků ze seznamu.

d. Třída MultiSelectSlicerTile
   Tento filtr je složen ze skupiny grafických elementů známých jako ToggleButton. Jde tedy o spojení tlačítka a CheckBoxu. Tlačítko si po zmáčknutí uchovává změněný stav, až do chvíle, kdy je znovu stlačeno.

e. Třída MultiSelectTreeTile
   Tento typ filtru je nejsložitější. Tento typ filtru se skládá ze stromové struktury dat, podle kterých je možno filtrovat. Vzhledem ke složitosti a množství jednotlivých prvků, podle kterých je možno vyhledávat má tento filtr speciální lištu, která umožňuje současné nastavení potvrdit nebo zrušit. Dále se na liště vyskytuje speciální tlačítko, které umožňuje vybrat nebo zrušit vybrání všech prvků. Třída oproti jiným filtrům obsahuje rekurzivní metody pro průchod stromem a také speciální způsoby označování prvků.

```
/// <summary>
/// Recursively searches item and all it's childrens
/// </summary>
/// <param name="item">Current Item</param>
/// <param name="Selector">Function called on each item</param>
/// <returns>List of TreeViewItems</returns>
private List<TreeViewItem> GetTreeViewItems(TreeViewItem item, Predicate<TreeViewItem> Selector)
{
    List<TreeViewItem> output = new List<TreeViewItem>();
    if (Selector(item))
        output.Add(item);

    if (item.Items.Count == 0)
        return output;

    foreach (TreeViewItem child in item.Items)
    {
        List<TreeViewItem> tmp = GetTreeViewItems(child, Selector);
        if (tmp.Count != 0)
            output.AddRange(tmp);
    }

    return output;
}
```

Obrázek 20 - Ukázka kódu metody GetTreeViewItems

3.3.12 Třída PivotTableTile

Tato třída slouží pro graficky náročné tabulky, které se liší od tabulek zobrazovaných v rámci třídy ChartTile. Tabulky třídy PivotTableTile umožňují různé barvy buněk, dále otevírání nebo zavírání různých částí tabulky, a to jak řádků, tak sloupců.

a. Vnitřní třída PivotCell
   Tato třída ukládá všechna data o jedné buňce získané z API a také pozici řádku a sloupce v tabulce.

b. Setter vlastnosti Data
   Vlastnost Data, která ukládá získaná data z API o tabulce, ale taky hned po přiřazení data použije pro definování tabulky. Z počátku resetuje zobrazovací prvky do výchozího stavu. V dalším kroku nastaví instanci třídy Grid a následně data rozdělí do jednotlivých buněk.

c. Metoda GetCell
   Tato metoda vygeneruje buňku s výchozím nastavením. Pro vygenerování se používá třída TextBlock, která zobrazuje text. Ta je dopněna třídou Border, která určuje velikost a barvu ohraničení buňky. Nakonec se v metodě nastaví parametry třídy Grid pro určení pozice buňky v tabulce.

d. Metoda GetAlignedCell
   Metoda GetAlignedCell využívá výše popsanou metodu GetCell a přidává ji parametr pro zarovnání textu.

e. Metoda Expand_Tapped
   Kliknutím na část tabulky, která jde rozevřít se vyvolá událost, kterou zachytí tato metoda. Tato metoda si z prvního parametru, a to odesílatele události, vytáhne jeho data, která poté převede do objektu použitelného pro API. Další data získá ze struktury tabulky a pošle požadavek na API. Při kladné odpovědi se data deserializují a tabulka se aktualizuje.

3.3.13 Třída CustomDataPoint

Tato třída slouží pro ukládání různých dat používaných pro grafy. Mimo jiné ukládá data pro buňku nebo část grafu, barvu, název a hodnotu. Také má reference na definici grafu a os.

a. Metoda SetObject
   Metoda SetObject nastavuje všechny proměnné třídy.

3.3.14 Statická třída Connection

Tato třída slouží pro jednodušší komunikaci se servery API a UWP. Třída Connection obsahuje metody pro posílání požadavků na API, metody pro otevření a uzavření kanálu push notifikací a další pomocné metody a data potřebná ke komunikaci. Dále obsahuje výchozí hodnoty, které se použijí, pokud se uživatel přihlašuje poprvé a zobrazí výchozí data z demo serveru s API.

a. Metoda GetCorrectUri
   Tato pomocná metoda přidá k URI doménu, pokud není zadána.

b. Metoda GetDataAsync
   Tato metoda zavolá GetCorrectUri pro opravu URI a poté pomocí třídy HttpClient, která je součástí tříd z dynamických knihoven společnosti Microsoft, vytvoří a pošle požadavek na server. Metoda dále nastavuje vlastní hlavičky požadavku. První hlavička je „Authorization“, která se skládá ze jména a tokenu, získaného hašováním hesla a URL požadavku. Další hlavičkou je „If-Modified-Since“, která se nastavuje z důvodu automatického cachování požadavků třídou HttpClient, což je pro naše potřeby nežádoucí. Následně se asynchronně volá daná URI a odpověď serveru se vrací.

```
/// <summary>
/// Sends GET request with authorization header, asynchronously
/// </summary>
/// <param name="Uri">Uri for request</param>
/// <returns>Response content</returns>
public async static Task<string> GetDataAsync(string Uri)
{
    Uri = GetCorrectUri(Uri);

    string result = "";
    using (HttpClient client = new HttpClient())
    {
        client.DefaultRequestHeaders.TryAppendWithoutValidation("Authorization",
            string.Format("{0}:{1}", UserName, CalcHMAC(Uri, Password)));

        // to overcome cache
        client.DefaultRequestHeaders.IfModifiedSince = new DateTimeOffset(new DateTime(2000, 1, 1));

        using (var response = await client.GetAsync(new Uri(Uri)))
        {
            result = await response.Content.ReadAsStringAsync();
        }
    }

    return result;
}
```

Obrázek 21 - Ukázka kódu metody GetDataAsync

c. Metoda PostDataAsync
   Tato metoda je velmi podobná metodě GetDataAsync. Hlavním rozdílem je změna požadavku na typ POST. Ten je vyžadován určitými funkcemi API, které potřebují zaslat data. Metoda PostDataAsync oproti předchozí metodě používá navíc třídu HttpStringContent, která data ve formátu JSON a znakové sadě UTF-8 upraví pro použití v API.

d. Metoda PutDataAsync
   Tato metoda je prakticky identická metodě PostDataAsync. Jediným rozdílem je změna požadavku na typ PUT.

e. Metoda CreatePushNotificationChannel
   Tato metoda pomocí předpřipravených tříd vytvoří kanál pro push notifikace a zaregistruje metodu, která bude obsluhovat požadavky na zobrazení push notifikace. Pokud byl kanál úspěšně vytvořen, uloží se jeho URI do nastavení a odešle se na API.

f. Metoda ClosePushNotificationChannel
   Tato metoda uzavře otevřený kanál push notifikací. Následně zruší URI kanálu z nastavení.

g. Metoda CalcHMAC
Metoda CalcHMAC slouží k zahašování autorizační hlavičky požadavku na API. Metoda funguje tak, že zpočátku heslo rozdělí na jednotlivé byty. Poté se vytvoří buffer z bytů hesla, který se zahašuje pomocí algoritmů HMAC a hašovacího algoritmu MD5. Do procesu se přidá URI požadavku a výsledný haš je vrácen jako řetězec typu base64.

```
/// <summary>
/// Encrypts password
/// </summary>
/// <param name="src">Uri</param>
/// <param name="password">Password</param>
/// <returns>Crypted password</returns>
private static string CalcHMAC(string src, string password)
{
    src = src.ToUpper();
    byte[] secretBytes = Encoding.UTF8.GetBytes(password);

    MacAlgorithmProvider alg = MacAlgorithmProvider.OpenAlgorithm(MacAlgorithmNames.HmacMd5);
    IBuffer buff = CryptographicBuffer.CreateFromByteArray(secretBytes);
    CryptographicHash hash = alg.CreateHash(buff);
    hash.Append(CryptographicBuffer.ConvertStringToBinary(src, BinaryStringEncoding.Utf8));
    IBuffer hashedValue = hash.GetValueAndReset();
    return CryptographicBuffer.EncodeToBase64String(hashedValue);
}
```

Obrázek 22 - Ukázka metody CalcHMAC

h. Metody SaveFilter
Tyto metody slouží pro uložení stavu filtrů a dynamického filtru na API serveru.

i. Metoda SendChannelToServer
Tato metoda získá hardwarový token, který je specifický pro každé zařízení. Jeho id poté překóduje do base64 řetězce. Ten je spolu s URI kanálu a dalšími údaji odeslán na API. Z API je od té chvíle pomocí URI kanálu možno odesílat push notifikace na zařízení.

j. Metoda OnPushNotification
Tato metoda reaguje na událost vyvolanou příchodem push notifikace. Pokud jde o určitý typ notifikace je navíc aktualizováno nastavení notifikací.

k. Metoda GetAndSortNotification
Tato metoda z API získá seznam notifikací, který je deserializován a následně seřazen podle data vyslání.

3.3.15 Třída General

Tato třída obsahuje pomocné funkce pro další třídy v projektu a také funkce pro uložení nastavení.

a. Vnitřní třída SettingId
Tato třída pouze ukládá klíče pro ukládání nastavení.

b. Metody GeneratePath
Tyto metody dynamicky vytvoří objekt třídy Path, který se používá k zobrazení vektorových dat jako jsou obrázky.

c. Metoda LoadSettings
Metoda zpočátku nastaví výchozí nastavení a poté se dívá, jestli existuje soubor s uloženým nastavením. Pomocí PasswordVault, který zajišťuje bezpečné uložení přihlašovacích údajů se získají přihlašovací údaje a z lokálního nastavení se získá uložená URI domény.

d. Metoda SaveSettings
Tato metoda ukládá veškerá nastavení pro připojení k serveru a autorizaci - jméno, heslo a URI serveru.

e. Metoda SaveCredentials
Metoda SaveCredentials ukládá do takzvaného PasswordVaultu přihlašovací údaje. Toto má zajistit, aby se údaje ukládaly bezpečně a nešly jednoduše nalézt v zařízení.

f. Metody ChangeUsername a ChangePassword
Tyto metody slouží pro uložení změny pouze jména nebo pouze hesla. Pro správnou funkci musí získat data z PasswordVaultu, nahradit dotyčný parametr a znovu uložit.

g. Metoda ChangeDomain
Tato metoda pouze v lokálním nastavení změní URI domény.

h. Metoda ConvertStringToColor
Tato metoda převádí barvu v řetězci ve formátu šesti hexadecimálních čísel na třídu Color. Metoda rozděluje hexadecimální číslice na části určené pro jednotlivé barvy a poté části převádí na decimální částice, které se dále používají.

i. Metoda ConvertIntToColor
Tato metoda převádí barvu zakódovanou ve skalární proměnné typu integer. Metoda funguje tak, že proměnnou rozdělí na tři části po jednom bytu, kde každý byte reprezentuje jednu barvu.

j. Metoda ParseHexadec
Tato metoda převádí hexadecimální číslo na decimální.

3.3.16 Třída Serialization

Tato třída slouží k serializaci a deserializaci všech dat. Data jsou serializována z objektů do řetězce typu JSON a naopak. Metody této třídy využívají obecných typů pro vstup a výstup.

a. Metoda GetObjFromJSON
Tato metoda převádí řetězec typu JSON na objekt.

b. Metoda GetJSONFromObj
Tato metoda naopak převádí objekt na řetězec typu JSON.

3.3.17 Třída FrameParameter

Tato třída slouží jako parametr, který se přenáší mezi stránkami zpracovávající dynamické navigační menu nebo zobrazení s grafy a tabulkami. Tohoto parametru se nejvíce využívá, když využíváme tlačítko zpět. Instance této třídy může obsahovat další instance sebe sama, a proto se můžeme nořit libovolně hluboko.

4 Závěr

Aplikaci se úspěšně podařilo dokončit se všemi popsanými náležitostmi v zadání. Realizovány jsou všechny požadované obrazovky. S použitím push notifikací operačního systému Windows jsou všechny notifikace okamžitě zobrazovány a uživatel tak má vždy aktuální informace. Layouty a grafy jsou načítány a zobrazovány podle potřeb uživatele a informací ze serveru. Filtry ukládají svůj stav lokálně i na server a tím umožňují rychlý návrat k požadovaným filtrovaným datům.

Literatura

[1] What's a Universal Windows Platform (UWP) app? - UWP app developer | Microsoft Docs. [online]. Poslední revize 22.března 2017 [citováno 16.dubna 2017] Dostupné z: https://docs.microsoft.com/en-us/windows/uwp/get-started/whats-a-uwp

[2] ALBAHARI, Joseph a Ben ALBAHARI. C# 6.0 in a nutshell: the definitive reference. Sixth edition. In a nutshell (O'Reilly & Associates). ISBN 978-1-491-92706-9.

[3] What is XAML? Learn to Develop with Microsoft Developer Network | MSDN [online]. Copyright © 2017 Microsoft [cit. 16.duben 2017]. Dostupné z: https://msdn.microsoft.com/en-us/library/cc295302.aspx

[4] Overview of the .NET Framework | MSDN. [online]. Ver. 110 [citováno 16.dubna 2017] Dostupné z: https://msdn.microsoft.com/en-us/library/zw4w595w(v=vs.110).aspx

[5] WinRTXamlToolkit Project Description [online]. Poslední revize 5.července 2016 [citováno 16.dubna 2017] Dostupné z: https://github.com/xyzzer/WinRTXamlToolkit/blob/master/ReadMe.md

[6] Introducing JSON | json.org [online]. [citováno 16.dubna 2017] Dostupné z: http://www.json.org/

Přílohy

Příloha na CD/DVD.

Obsah CD/DVD:

- NavodKAplikaci.pdf – návod k online i offline instalaci a nastavení aplikace pro správnou funkci
- ZdrojovyKod.zip – zdrojové kódy programu
- OfflineInstalator.zip – soubory pro offline instalaci